UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 1559

Programme 4
Robotique, Image et Vision

LOCAL EDGE GROUPING BY SIMPLE PROCESS ITERATION

Frank MANGIN
Marc BERTHOD
Josiane ZERUBIA

Novembre 1991

RR_1559

Local edge grouping by simple process iteration

Frank Mangin
Marc Berthod
Josiane Zerubia
INRIA Sophia Antipolis
2004, Route des Lucioles
F-06565 Valbonne Cedex, France
Tel: (33) 93-65-78-66 Fax: (33) 93-65-77-66
e-mail: mangin@sophia.inria.fr
berthod@sophia.inria.fr zerubia@sophia.inria.fr

Abstract: A new iterative algorithm for edge intensity images enhancement is proposed. It uses local cooperation-inhibition processes to produce an edge image in which the most relevant contours have reached maximal activation, and small gaps and junctions have been filled in. Implementation on a massively parallel machine provided high speed performances, and results of experimentations with images of real scenes are presented. The algorithm is robust in complex edge images context, and is perfectly stable under any number of iterations.

Keywords: early vision, contour grouping, contour detection, iterative parallel algorithms

---

Un algorithme d'itération de processus élémentaires pour le groupement de contours

Résumé : Un algorithme itératif pour l'amélioration des images de contours est décrit. Il utilise des processus locaux de coopération - inhibition pour produire une image de contours dans laquelle les contours les plus pertinents ont atteint le niveau maximal d'activation, et où les petites interruptions et les jonctions sont complétées. L'approche retenue a permis une réalisation très efficace sur machine parallèle à grain fin, et des résultats pour des images de scènes réelles sont présentés. Appliqué à des images de contours complexes et denses, le modèle s'est révélé très robuste et parfaitement stable quel que soit le nombre d'itérations.

Mots clés : vision préattentive, détection de contours, groupement de contours, algorithmes itératifs parallèles

Contents

1 Introduction

Contour detection in digital image processing is a fundamental step towards scene interpretation. A large class of methods, most widely used to extract image contours, consists in applying to each pixel a local discontinuity detector filter, and in thresholding the result to produce an edge map.

The information extracted this way is difficult to use directly for scene analysis, because there is no straightforward relationship between digital image intensities and contours or surfaces of the scene real objects. Though, the human visual system performs scene interpretation using only perceived light intensities. The first step of this interpretation is known as "Perceptual Grouping". During this preattentive process, regions are merged or splitted, new edges are inferred, some irrelevant edge elements disappear, following rules first studied by "Gestalt" psychophysiologists (see [9]). It should be emphasized that these processes precede the interpretation phase, and thus involve no "high-level" knowledge. This step of perceptual organisation is in our opinion a key condition for the success of interpretation.

In the following section, we propose a global model architecture for implementing perceptual contour grouping in the context of digital image processing. The algorithm described in section 3 is a first element of this architecture. Results of experiments with images of real scenes are presented in the last section.

2 General Model Overview

2.1 Constraints

Although toy images may be interesting to explain the behavior of an algorithm, they can't be used to validate it. Only real images can be used for this purpose: in all our experimentations, source images are digitalized video camera views. Moreover, since perceptual grouping involves no high level knowledge, the model should be able to handle any type of images: our set of examples is composed of both indoor, outdoor, and aerial views.

Finally, since the final purpose of an image processing algorithm is its implementation on real-time vision system, speed efficiency is an imperative requirement.

2.2 Proposed Approach

It is well known that grouping contours and selecting most salient ones requires long-range interaction between edge elements, as shown in Figure 1 where the best grouping can't be deduced from local information.

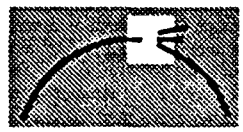

Figure 1: Need for long-range interactions

Several approaches can be used to implement these interactions. A diffusion phenomenon is obtained by iterating local updating of field elements in Markov Random Field models (see [3], [6]). Here a pixel value modification through one iteration only depends on a neighbourhood whose size is given by the order of the random field. As a consequence, the interaction rules between distant pixels are defined in an implicit way, which provides few control over their definition as long as a multi-scale approach is not used. Furthermore the distance of propagation is proportional to the number of iterations, which severely limits speed efficiency.

An other solution is to propagate information along the contours. This approach was used in [8] to detect salient contours. Here more accurate control over the interaction rules is provided, however the time needed for two pixels to interact is proportional to their distance along the contour.

The general framework we propose to use is a model of local interactions between objects, following rules depending on a predefined set of attributes attached to each object. So as to meet time-efficiency constraints and accurate definition of the interaction rules, we also believe that a multiscale approach should be used.

The algorithm described below is the first step towards this model. It implements local interactions for the lowest scale only, where objects are the pixels of an edge image. Five attributes have been used at this scale:

- the edge element intensity, which is updated at each iteration
- 2 caracteristic contour directions as defined below, used in a cooperation step
- two reference values computed over the pixel neighbourhood, used in an inhibition step.

The algorithm is obtained by iterating the following sequence:

- evaluation of attributes
- application of the updating rules.

Related works include relaxation labeling applied to contour enhancement (see [11]). In our case, using more complex rules lead to better gap filling capabilities as well as perfect stability for any number of iterations. Significant theoretical advances in computing curve orientation and curvature for curve inference has been made by Parent and Zucker in [7]. Their results should prove usefull for next elements of our model, when higher scale will require more elaborated curve segment modelisation. The cooperation-inhibition concept has also been used by Grossberg and Mingolla (see [4]). In that work, iteration of local interactions allowed to implement complex contour grouping rules observed in human vision. However working with more complex images coming from real world scenes lead us to use simpler and thus more robust rules.

Neighbourhoods of radius 4 were used to compute the attributes. Below this size, contour elements are correctly caracterized with only directional attributes. Bigger sizes would require more complex notions such as average curvature or length for example.

To achieve time efficiency constraints, the algorithm was designed to allow massively parallel computations, while using only three simple basic operations:

- convolution of the pixel neighbourhood by precomputed kernels.

- maxima extraction
- thresholding

The results of Section 4 show that this choice lead to very high speed performances.

3 Description of the Algorithm

The algorithm starts with images of edge intensities. They can be obtained for example from an edge or line detection process applied on a grayscale image, followed by a no-maxima suppression in the contour direction. The final image is obtained by iteration of a two-step process over the image of edge intensities:

- Local cooperation: at positions where an incoming contour has been detected, compute a best continuation direction, and update neighbouring edge intensities accordingly.
- Inhibition: a no-maxima suppression is performed at each position, in the "maximal reinforcement direction" defined below.

Each step is detailed in the following sections. The result, composed of the final edge intensities, is obtained in less than ten iterations, without any post-processing.

Two parameters control the algorithm:

- an "activation threshold" M_a, which selects the positions that take part in the cooperation step.
- a "continuation threshold" M_c (see below).

The computations involving orientation use a sample of 16 predefined directions with angles $\theta_k = 2k\pi/16, k = 1,..,16$. These 16 directions define 8 orientations $\psi_k = \theta_k \bmod \pi = \theta_{k+8} \bmod \pi$. In the following sections, the terms "orientation" and "direction" always refer to these definitions.

The pixel value at position (i,j) is denoted by I_{ij}.

$\mathcal{V}_n(i,j) = \{(u,v)|(|u-i| \leq n)$ and$(|v-j| \leq n)\}$ denotes the square neighbourhood of radius n of (i,j).

3.1 Local Cooperation

For each "active" edge element (i.e. such that $I_{ij} > M_a$), a "support value" L_{ijk} is first computed for each of the 16 directions. The maximum is extracted, and is defined to be the "incoming direction". The remaining support values are then weighted according to this incoming direction, so as to favor straight continuations. A second maximum is extracted over these modulated values, and if it is greater than the continuation threshold M_c, it defines the "continuation direction".

The "incoming direction", and the "continuation direction" if it is defined, are then used to update the intensities of the neighbouring edge elements.

3.1.1 Computing the Edge Intensities

16 convolution kernels are defined for this purpose. Lateral inhibition is used to avoid selecting a high intensity orthogonal contour instead of a low intensity but well-oriented one. So the kernels have a positive center and a negative surround along the direction of interest, and values decrease with the distance:

$$\Gamma_{uvk} = \begin{cases} 0 & \text{if } \begin{pmatrix} sin\theta_k \\ cos\theta_k \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} \leq 0 \\ (d_0^2 - d_2^2)\exp(-|d_2|^5/5 - d_1^2/9) & \text{otherwise} \end{cases}$$

where $d_1 = \sqrt{u^2 + v^2}$

d_2 is the distance between point (u, v) and line θ_k,
and d_0 is the distance below which the point contribution is positive:

$$d_2 = d_1 \sin(\arctan(v/u) - \theta_k)$$
$$d_0 = 0.9$$

The support values are thus defined as follows:

$$L_{ijk} = \sum_{(u,v)\in V_4(i,j)} I_{uv}\Gamma_{u-i,v-j,k}$$

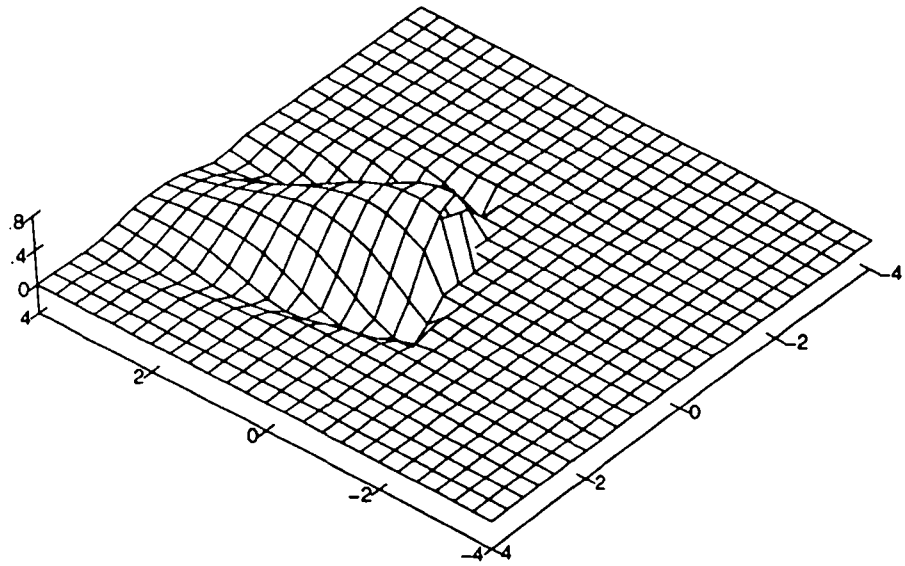

Figure 2: Edge intensity kernel, $\theta = \pi/8$

3.1.2 Extraction of Incoming and Continuation Directions

For each active position, the incoming direction k_{ij}^{in} is the one that maximizes the support value:

$$L_{ijk_{ij}^{in}} = \max_k L_{ijk}$$

The support values are then weighted according to the incoming direction. Since the contour detected in the incoming direction will produce responses in the surrounding directions as well, and since we don't want this contour to be detected twice, the weights for the directions near $\theta_{k_{ij}^{in}}$ are set to zero; the weights are set to 1 for a straight continuation and to 0.5 for an orthogonal continuation, which has proven to provide the best results in our experiments. This yields:

$$L'_{ijk} = \mu(k_{ij}^{in}, k) L_{ijk}$$

$$\text{with} \quad \mu(k_1, k_2) = \begin{cases} 0 & \text{if } |k_1 - k_2| \leq 3 \\ 1 - |k_2 - ((k_1 + 8) \bmod 16)|/8 & \text{otherwise} \end{cases}$$

The continuation direction k_{ij}^{out} is then extracted from these weighted support val-

ues:

$$L'_{ijk_{ij}^{out}} = \max_k L'_{ijk}$$

It is considered undefined if

$$L'_{ijk_{ij}^{out}} < M_c$$

M_c is thus the threshold below which a position should be regarded as a real end-point. This method implements a trade-off between the intensity of the continuation contour, and its angle with the incoming contour.

3.1.3 Updating

Since most positions lie on already well-established contours, we don't want them to create new active positions. This is why end-points are first detected: a position is defined to be an end-point if among its 8 nearest neighbours, only one has a non-zero edge intensity, or two have non-zero intensities and are connected in a 4-connexity pattern. In the example of Figure 3, white pixels are end-points. A position whose intensity is zero before the cooperation can be updated only by end-points.

Figure 3: Definition of end-points

Let $\mathcal{P} = \{(u,v) \mid I_{uv} > 0\}$ be the set of positions with non-zero intensities before any updating has taken place. Each active position (i,j) at which a continuation direction k_{ij}^{out} has been found updates its neighbouring positions as follows:

$$\begin{aligned}
&\forall (u,v) \in \mathcal{V}_2(i,j) \\
&\text{if } (u,v) \in \mathcal{P} \text{ or } (i,j) \text{ is an end-point} \\
&I_{uv} \mathrel{+}= W_{ij}\Gamma_{u-i,v-j,k_{ij}^{in}} + W_{ij}\Gamma_{u-i,v-j,k_{ij}^{out}}
\end{aligned}$$

Notice that the neighbourhood size is smaller than the one used to compute the support values, and that the updating amount only depends on the intensity of the updating pixel. In particular, it doesn't depend on the previous value of I_{uv} or on the support values $L_{ijk_{ij}^{in}}$ and $L'_{ijk_{ij}^{out}}$, thus defining a so-called "winner-take-all" update rule.

Since there are inhibition terms in the contour kernels, the intensities obtained after the cooperation step are set to zero when negative.

3.2 No-maxima Suppression

After the cooperation step, which acts as a hypothesis generator, we have to select the optimal contour positions. No-maxima suppression along the gradient orientation has often been used for this purpose in other models. However using a "contour orientation" leads to eliminate junction edge elements, as shown in Figure 5.

A quite natural approach successfully solves this problem: let us define the "maximal reinforcement orientation" Φ_{ij} as the orientation for which an edge element received its maximal contribution in the cooperation step:

$$\Phi_{uv} = \theta_{k^0_{uv}} \bmod \pi$$

$$\text{with} \quad k^0_{uv} = k^{in}_{i_0 j_0} \quad \text{or} \quad k^m_{uv} = k^{out}_{i_0 j_0}$$

$$\text{and} \quad W_{i_0 j_0} \Gamma_{u-i_0, v-j_0, k^0_{uv}} = \max_{(i,j) \in \mathcal{V}_4(u,v)} \left[\max_{k \in \{k^{in}_{ij}, k^{out}_{ij}\}} W_{ij} \Gamma_{u-i, v-j, k} \right]$$

This maximal reinforcement orientation is in some way the reason why a position has a high intensity. The key idea is to set the intensity of a position to zero only if a neighbouring position obtained a higher intensity *for the same reason*. A reference value is computed on each side of a position orthogonally to the maximal reinforcement orientation. One or two pixels of $\mathcal{V}_1(i,j)$ are used on each side, as shown in Figure 4.

Let J_{ij} and J'_{ij} be the reference values, we have:

$$J_{ij} = (\delta\Phi(i,j,u_1,v_1) I_{u_1 v_1} + \delta\Phi(i,j,u_2,v_2) I_{u_2 v_2})/2$$

$$\text{with} \quad \delta\Phi(i,j,u,v) = \begin{cases} 1 & \text{if } \Phi_{ij} = \Phi_{uv} \\ 0 & \textit{otherwise} \end{cases}$$

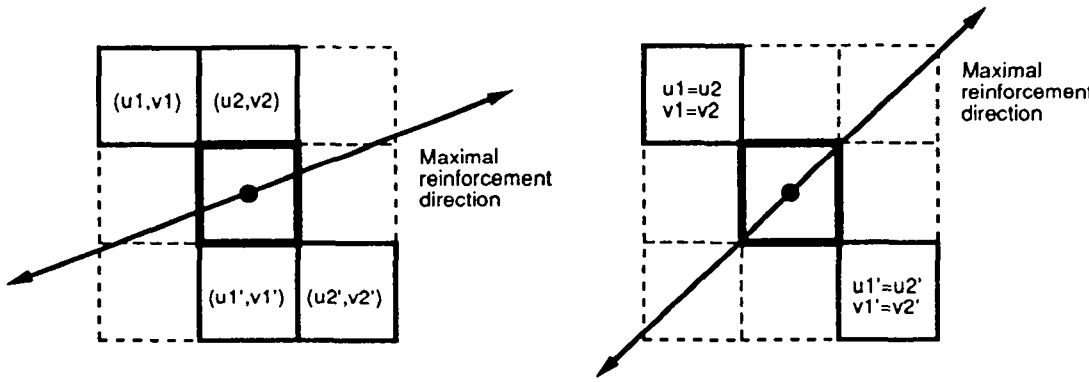

Figure 4: Neighbours used to compute reference values

and the same for J'_{ij}.

The edge intensities are then updated by simply setting to zero those which are less than one of their reference values:

$$\forall(i,j) \mid (I_{ij} < J_{ij}) \text{ or } (I_{ij} \leq J'_{ij})$$
$$I_{ij} = 0$$

Finally, a global thresholding is performed on the image to maintain the edge intensities below 1:

$$\forall(i,j)\ I_{ij} = \min(I_{ij}, 1)$$

The resulting image of edge intensities forms the input of the cooperation step of the next iteration.

3.3 Convergence and Stabilisation

One remarkable feature of the algorithm is that the resulting image remains very stable under any number of iterations. This has been achieved by strictly selecting the edge elements that take part in the cooperation step. Figure 14 shows a resulting image after 100 iterations.

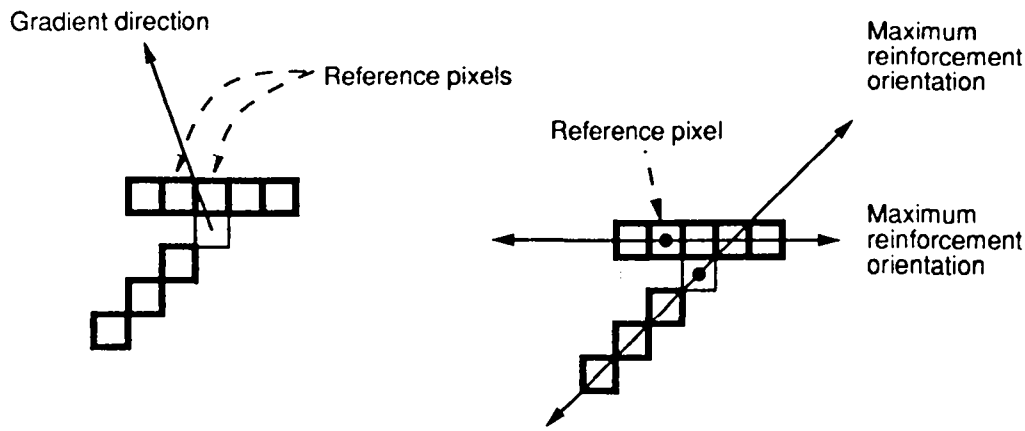

Figure 5: Effect of no maxima suppression on junction pixels

Interesting groupings however are obtained in far less iterations, and all the results presented have been obtained after 10 iterations.

At any given active position, two cases may occur:

- a continuation contour is detected in the neighbourhood of radius 4: since the radius of the neighbourhood used for updating is 2, the gap will be filled in 2 or 3 iterations.

- if no continuation contour is detected, no updating takes place

This explains why very few iterations are required. More iterations are usefull only in one case: a contour formed by a long section of edge elements whose intensities are smaller than the activation threshold, and with some high intensity edge elements who will propagate the activation 2 pixels further at each iteration. Thus the choice of 10 iterations corresponds to contours with non-zero inactive sections of maximal length 20, which covers the majority of contours in most images. If many such contours are expected in an image, the activation threshold should be lowered.

4 Experimental Results

4.1 Implementation

The algorithm has been implemented on a Connection Machine 2 system with a Sun4/390 front-end. The program is written in C^*, a parallel extension of C for the Connection Machine. The pixel values, edge intensities, and kernel terms all are 32 bits floating-point values.

The source code has really not been optimized. For example about half of the terms in each convolution kernel are zero, and the corresponding floating-point multiplications could easily be avoided by using precomputed masks. Moreover, pixels that reached saturation value 1 are still updated, which is useless. Even so, 10 iterations are performed in about 3 seconds when using one processor per pixel in the image (e.g. 128x128 image on the 16k processors Connection Machine of Inria). This level of performance was achieved by limiting the algorithm to very basic operations, as described in Section 2.2. Writing the program in a lower-level language (like C-Paris for the Connection Machine) with some optimization should make the algorithm at least twice as fast.

4.2 The Source Images

Three grey-level images were used in our experiments:

- Image 1 is a 256x256 indoor view (Figure 6)
- Image 2 is a 512x512 outdoor road scene (Figure 10)
- Image 3 is a 256x256 satellite image (Figure 15)

A step edge detector has been applied to images 1 and 2. Deriche algorithm, a recursive implementation of filters derived from Canny criteria ([1],[2]), has been used for this purpose. Classical no-maxima suppression in the gradient direction was then applied to form the algorithm input images. In Figures 7, 8, 11 and 12 thresholding has been used to obtain good legibility of the printed images.

In order to address the problem of road detection, a white ridge detector was applied to the satellite image. This detector, proposed by Ziou ([10]) has optimal response for thin white lines in a black surround. A no-maxima suppression was then applied as proposed in [10] to obtain the initial edge intensity image (Figures 16 and 17).

4.3 Evaluation of the Results

The results presented in Figures 9, 13 and 18 were all obtained after 10 iterations, and are the exact output of the algorithm without any thresholding, although low intensities are invisible in printed images.

The activation threshold was set to 0.05 for images 1 and 2, and to 0.2 for image 3. With these values, about 25% of non-zero edge elements were active in images 1 and 2, and 15% in image 3. The continuation threshold M_c was set to 0.025 for images 1 and 2, and to 0.05 for image 3.

The relevant contour elements all reached saturation value 1, while the value of isolated ones remained constant, and no thick lines have been introduced by the cooperation process. Most junctions and small gaps were successfully completed, as long as their size is less than the size of the interaction window. Longer gaps have been interpreted as contour end-points, which is the relevant choice at this scale, and no further diffusion has taken place at the end-points.

The text at the top of image 2, where many contours are only one pixel apart, demonstrates the robustness of the process in a dense edge image context (see Figures 11, 13).

5 Conclusion

A new iterative algorithm that exhibits powerfull contour enhancement capabilities has been described, and has been successfully applied to complex images of real scenes. It achieves very high speed efficiency, while being perfectly stable under any number of iterations. This algorithm applied to edge images greatly improves the results of subsequent contour chaining or recognition processes.

However it is part of a more global architecture, which is currently being implemented. This multi-scale model should provide an efficient implementation of complex rules for long-range interactions, which are required for perceptual contour grouping.

References

[1] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8(6):679–698, november 1986.

[2] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, pages 167–187, 1987.

[3] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6(6), november 1984.

[4] S. Grossberg and E. Mingolla. Neural dynamics of perceptual grouping: Textures, boundaries, and emergent segmentations. *Perception & Psychophysics*, Vol. 38-2:141–171, 1985.

[5] S. Lehar, T. Howells, and I. Smotroff. Application of Grossberg and Mingolla neural vision model to satellite weather imagery. In *Proc. INNC 90*, pages 805–808, Paris, 1990.

[6] J. L. Marroquin. A markovian random field of piecewise straight lines. *Biological Cybernetics*, 61:457–465, 1989.

[7] P. Parent and S. W. Zucker. Trace inference, curvature consistency, and curve detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-11(8), Vol. august 1989.

[8] A. Sha'ashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *Proc. NIPS 90*, 1990.

[9] M. Wertheimer. Untersuchungen zur lehre von der gestalt ii. *Psychol. Forsch.*, (4), 1923. Translated as *Principles of Perceptual Organization*, in *Readings in Perception*, David Beardslee and Michael Wertheimer, Eds., (Princeton, N.J), 115-135, 1958.

[10] D. Ziou. Line detection using an optimal IIR filter. *Pattern Recognition*, Vol. 24(6):465–478, 1991.

[11] S. W. Zucker, R. A. Hummel, and A. Rosenfeld. An application of relaxation labeling to line and curve enhancement. *IEEE Transactions on Computers*, Vol. C-26(4), april 1977.

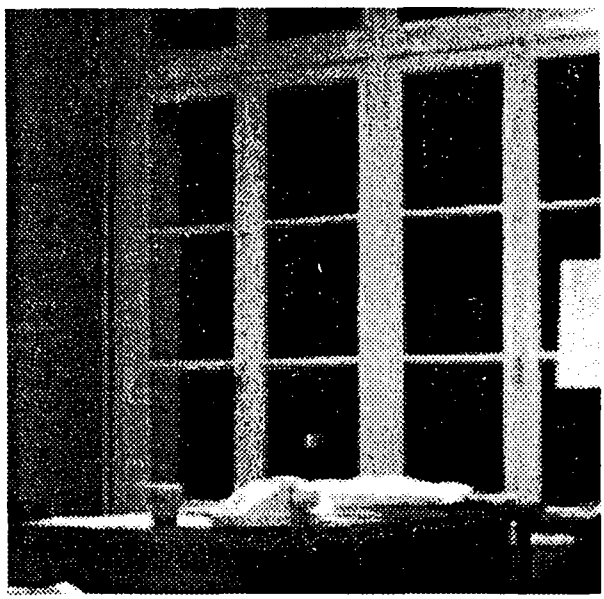

Figure 6: Source image 1

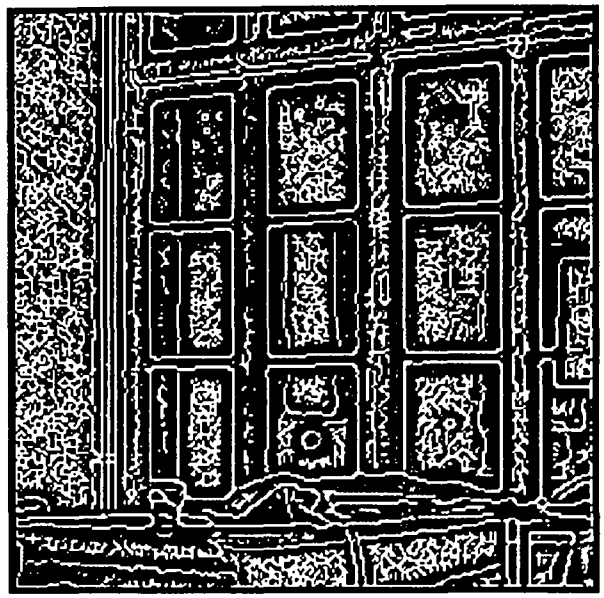

Figure 7: All edges detected after edge detection + no-maxima supression

Figure 8: Active edges (M_a=0.05) after edge detection + no-maxima supression

Figure 9: Resulting image after 10 iterations

Figure 10: Source image 2

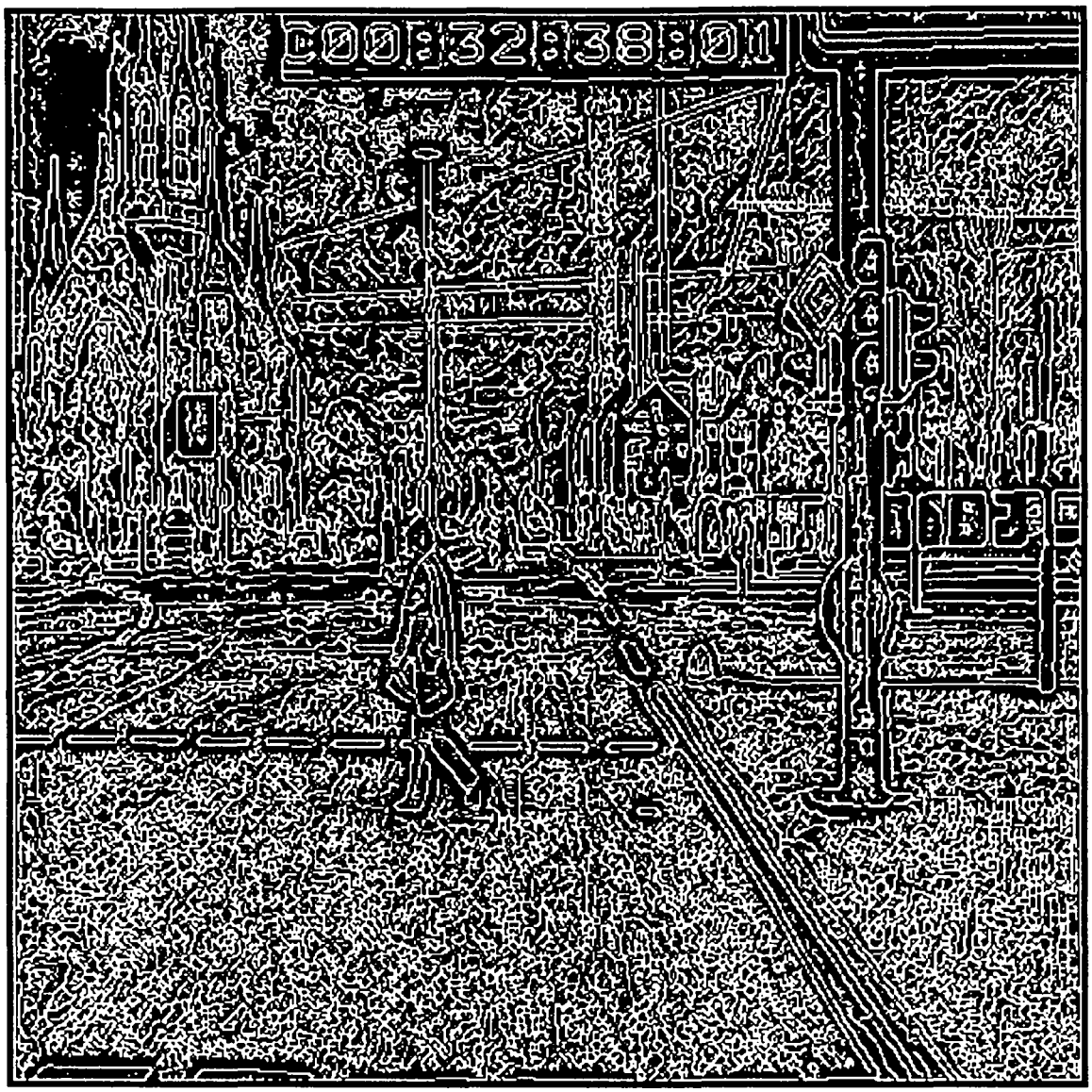

Figure 11: All edges detected after edge detection + no-maxima supression

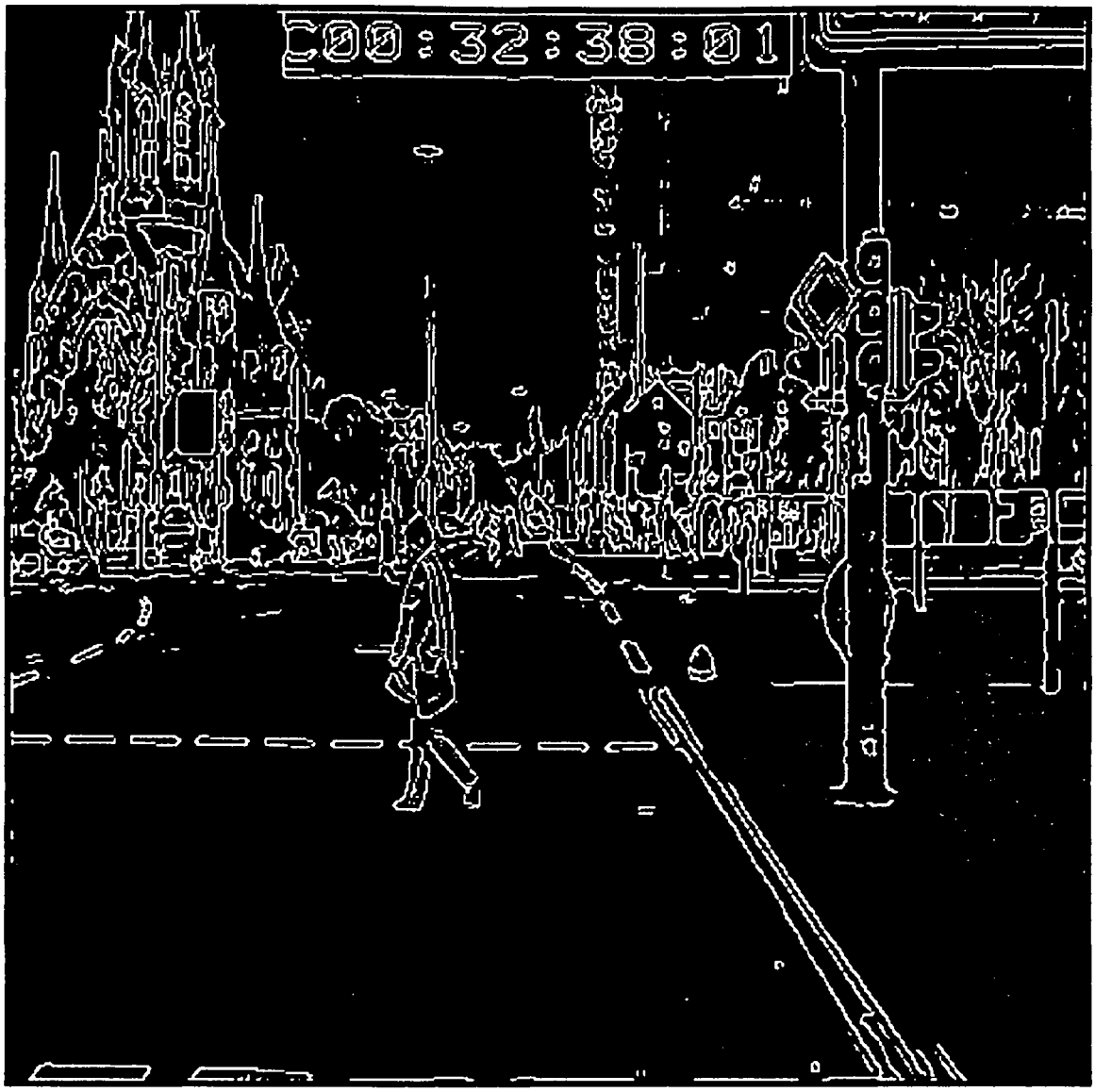

Figure 12: Active edges (M_a=0.05) after edge detection + no-maxima supression

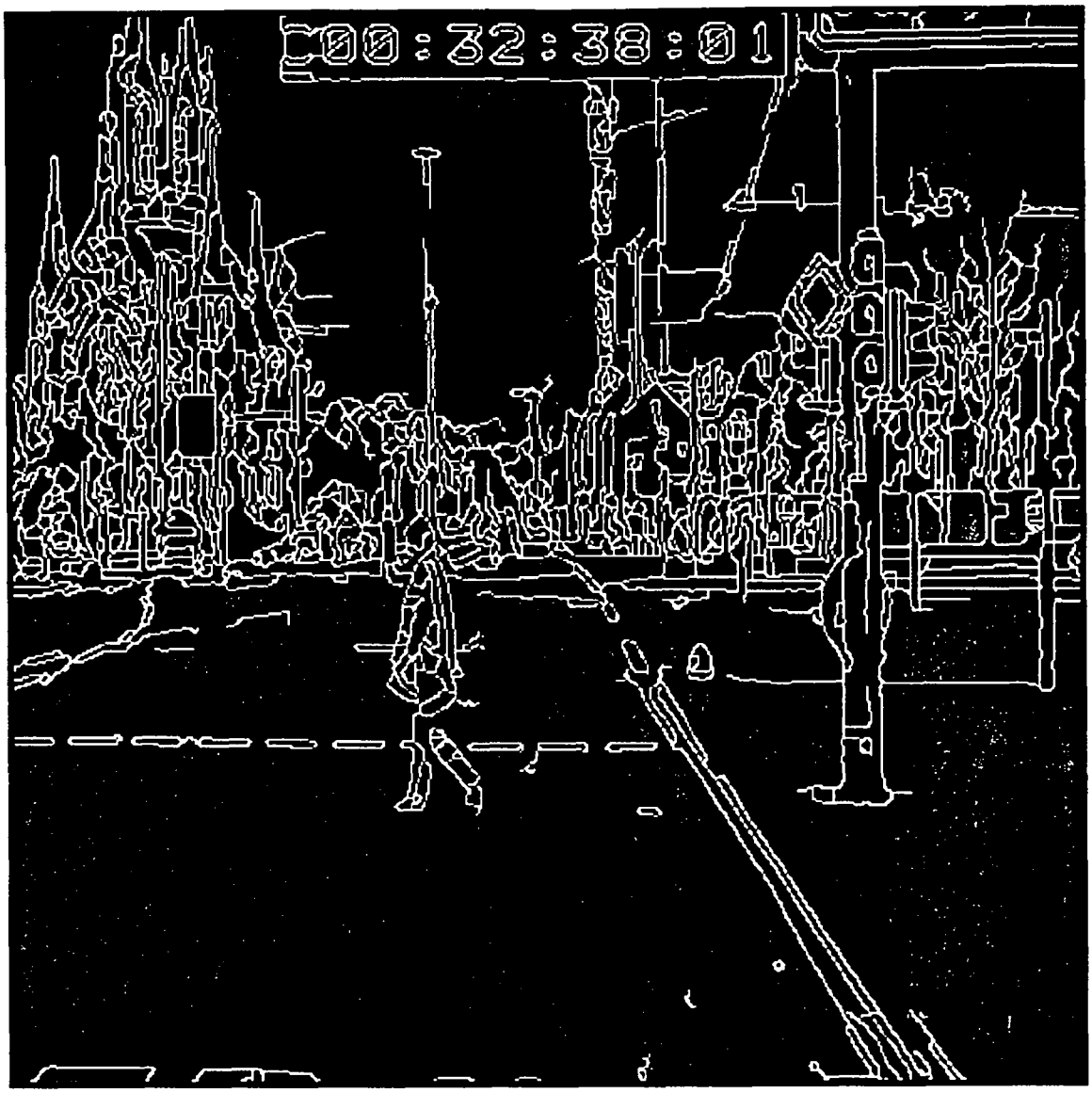

Figure 13: Resulting image after 10 iterations

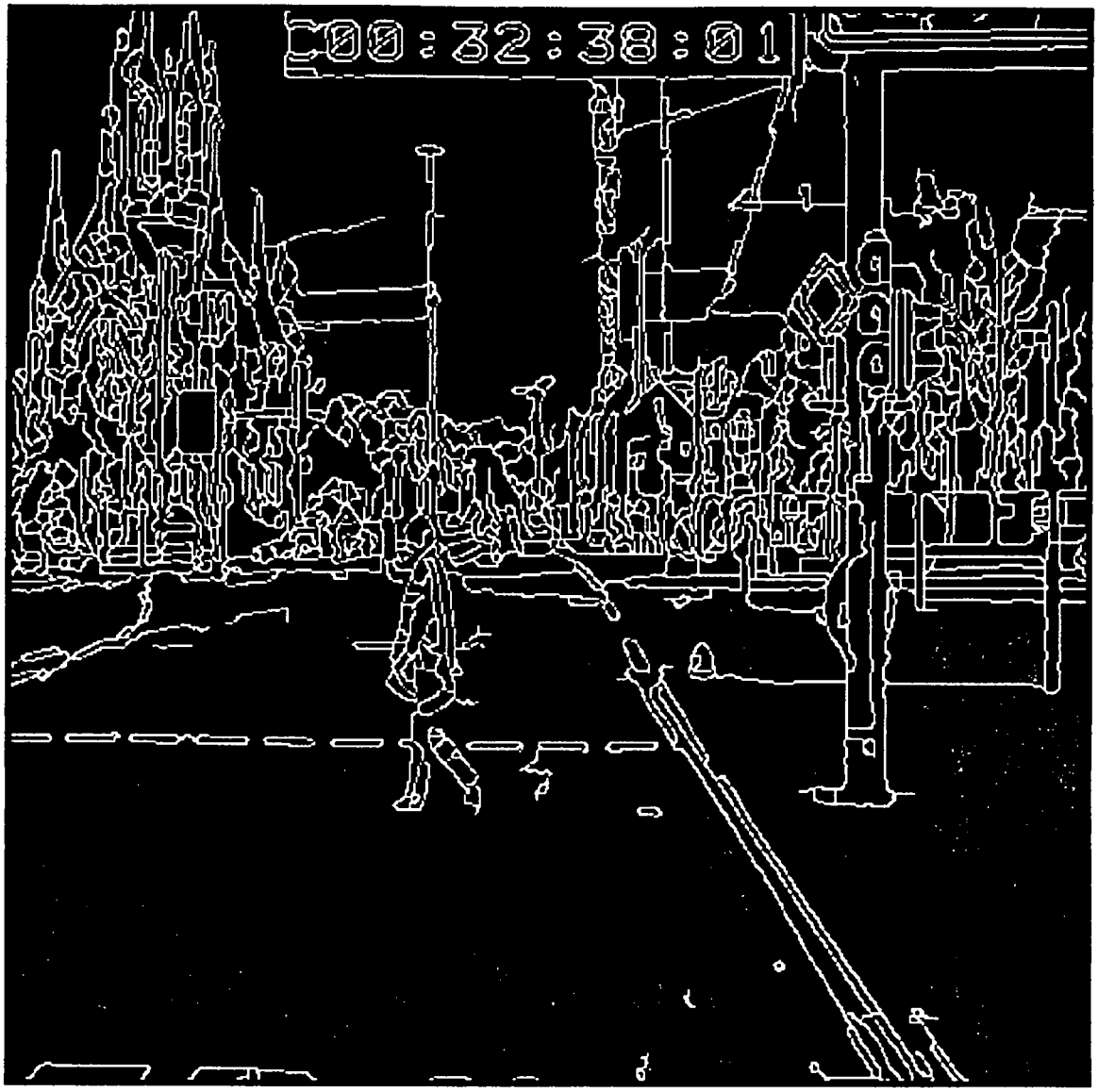

Figure 14: Resulting image after 100 iterations

Figure 15: Source image 3

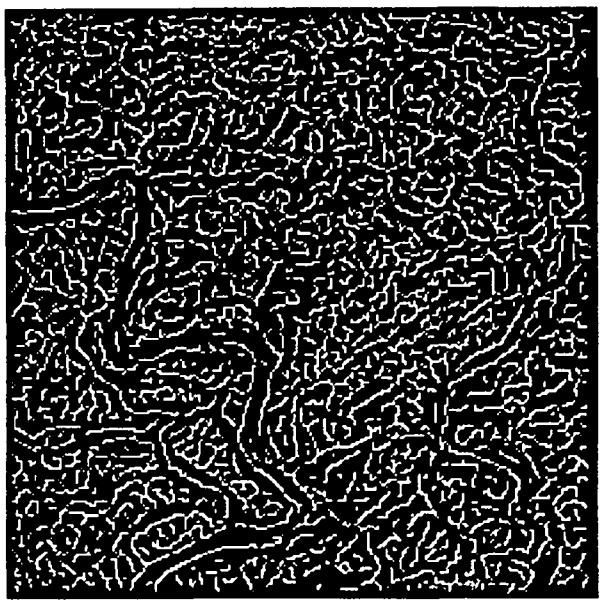

Figure 16: All the edges detected after white ridge detection + no-maxima supression

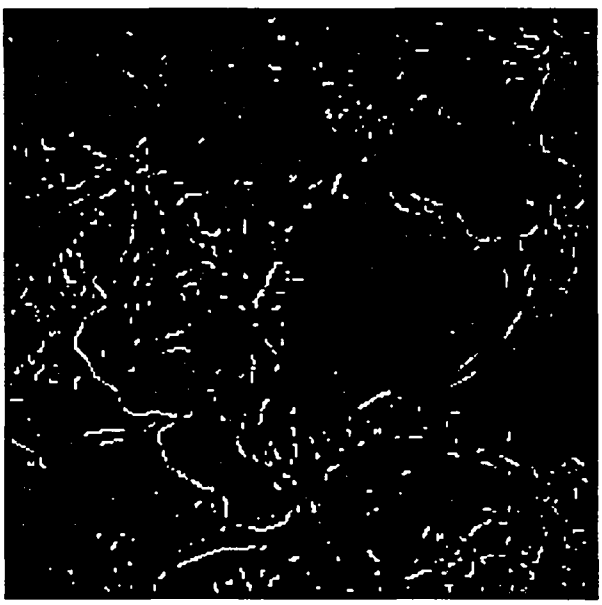

Figure 17: Active edges (M_a=0.2) after white ridge detection + no-maxima supression

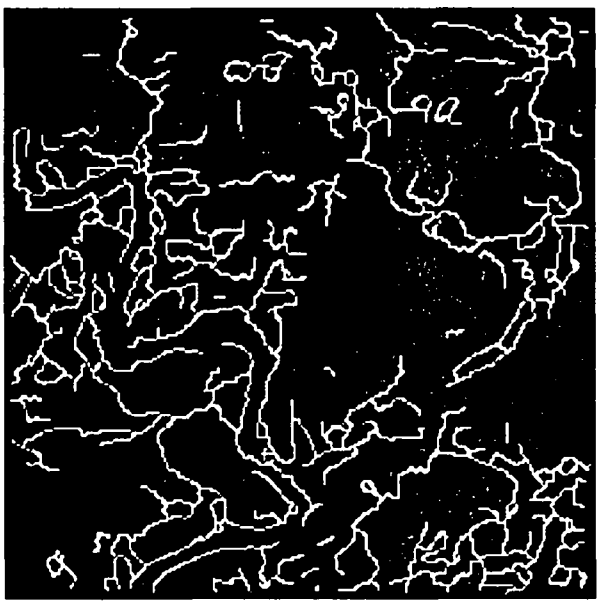

Figure 18: Resulting image after 10 iterations

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique.

ISSN 0249-6399